#20
D. Board
12/11/02

219.36435X00
Serial No. 09/215,788

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE BEFORE THE BOARD OF APPEALS AND INTERFERENCES

Appellants:	Jerrie L. COFFMAN et al.
Application No.:	09/215,788
Filing Date:	December 21, 1998
Title:	EFFICIENTLY EXPORTING LOCAL DEVICE ACCESS ONTO A SYSTEM AREA NETWORK USING A DIRECT-CALL INTERFACE
Art Unit:	2152
Examiner:	B. Prieto

APPEAL BRIEF

Assistant Commissioner of Patents
Washington, D.C. 20231

December 6, 2002

Sir:

Pursuant to Appellants' earlier filed Notice of Appeal dated on September 16, 2002, Appellants hereby appeal to the Board of Appeals and Interferences from the decision (Paper No. 12) dated on June 19, 2002 with respect to all the finally rejected claims 1-28.

This Appeal Brief is being filed in triplicate.

I. Statement of Real Party in Interest

Pursuant to 37 C.F.R. §1.92(c)(1)(as amended), the real party in interest is:

Intel Corporation
2200 Mission College Blvd., SC4-202
Santa Clara, CA 95052

II. Related Appeals and Interferences

Pursuant to 37 C.F.R. §1.192(c)(2)(as amended), although the real party in interest has other pending appeals and interferences, none of the other pending appeals and interferences is believed to directly affect or be directly affected by, or to have any bearing upon the decision of the Board of Patent Appeals and Interferences in this appeal.

III. Status of the Claims

Claims 1-21 and 23-28 are pending in this application at the filing of this Brief. Independent claim 22 has been canceled without prejudice or disclaimer[1]. Claims 1-18 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Heil et al., U.S. Patent No. 6,173,374, as modified to incorporate selected features from the "Intelligent I/O (I_2O) Architecture

[1] See Amendment After Final filed concurrently with this Appeal Brief.

Specification," Version 1.5, March 1997.[2] Separately, claims 19-21 and 23-28 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Heil et al., U.S. Patent No. 6,173,374, as modified to incorporate selected features from the "Intelligent I/O (I_2O) Architecture Specification," Version 1.5, March 1997, and Bonola, U.S. Patent No. 6,321,279. Appellants are appealing from the outstanding rejections of claims 1-21 and 23-28. Of the appealed claims, claims 1, 7, 14 and 23 are independent claims.

IV. Status of the Amendments

An Amendment after final under 37 C.F.R. §1.116(b) was filed on June 28, 2002, but will not be entered upon filing of this Appeal Brief.[3] A second Amendment after final under 37 C.F.R.

---

[2] Claims 1-28 were previously finally rejected under 35 U.S.C. §103(a) as being unpatentable over Heil et al., U.S. Patent No. 6,173,374, as modified to incorporate selected features from Angelo et al., U.S. Patent No. 6,173,374 from which an Appeal Brief was filed on November 13, 2001. However, the Examiner has withdrawn the application from the Board of Appeals and Interferences, reopened prosecution and substituted Angelo et al., U.S. Patent No. 6,061,794, as a secondary reference, with the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, a cited reference expressly acknowledged on page 10 of Appellants' specification in order to support the rejection of claims 1-28 as pending on this appeal.

[3] In the Examiner Interview Summary (Paper No. 12) attached with the final Office action dated on June 19, 2002, the Examiner has expressly indicated if the limitation "wherein said Connection Manager queries said Local Transport so as to determine the number of input/output platforms (IOPs), builds an IOP descriptor structure for each input/output platform (IOP) which includes an exported table of function call pointers and the context required by the Local Transport to communicate with the input/output platform (IOP)" of dependent claim 19 is incorporated into

§1.116(b) has been filed concurrently with this Appeal Brief, canceling independent claim 22. As a result, only claims 1-21 and 23-28 are pending on this Appeal.

V. Summary of the Invention

One of inherent challenges of a system area network (SAN) cluster as shown in FIG. 1 which links remote servers 10-18 and network-connected storage devices 130 connected to a host server (local server) 10 is to design a data transport mechanism that can deliver and exchange a large amount of data messages quickly between end nodes in the SAN cluster.

Traditional data transports between end nodes (for example, host server 10 and remote servers 14-18) in a SAN cluster are done through the network infrastructure provided by an operating system (OS) of a host server (local server) 10. However, a large amount of system processing overhead and an extended processing time are required to process each data message between the host server 10 and remote servers 14-18 as shown in FIG. 2. This overhead limits input/output (I/O) bandwidth, increases input/output (I/O) latency, and increases the response time

---

each of independent claims 1, 7, 12, 19 and 22, all claims 1-28 will be allowed and the instant application will be in condition for issuance. However, Appellants believe the indication is entirely unnecessary since each of Appellants' independent claims 1, 7, 12, 19 and 22 is already deemed patentably distinguishable over all cited prior art of record, including the Examiner's proposed combination of Heil et al., U.S. Patent No. 6,173,374 and the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997.

to the application.[4]

Another data transport mechanism between end nodes is to use a unique application to generate special application-to-application messages to the remote node in order to access a remote server in a SAN cluster. A remote application running on the remote node in the SAN cluster must issue an input/output (I/O) request to the remote operating system (OS) installed at the remote server on behalf of the local application. This way the operating system (OS) overhead on the host server (local server) 300 may be avoided. However, there are still a great deal of undesirable coordination between cooperating applications of the host server 300 and the remote server in the SAN cluster.[5]

In order to avoid incurring the overhead of the operating system (OS) protocol stack and without coordinating special application-to-application messages between nodes in a system area network (SAN) cluster, a more compact and especially designed driver system is provided at a host server according to an embodiment of the present invention in order to advantageously enable a direct, transparent access to I/O storage devices connected to the host server within a SAN for efficient sharing of resources and databases among all network members.[6] Such a driver system

---

[4] See page 3, lines 18-23, and page 4, lines 1-2 of Appellants' specification.

[5] See page 4, lines 12-18 of Appellants' specification.

[6] See page 8, lines 10-15 of Appellants' specification.

may be split into two modules in accordance with the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997:[7] a host driver module 310 - an upper module which is a host OS-specific portion installed as part of an operating system (OS) of the host server 300; and a device driver module 322 - a lower module which is a device-specific portion that interfaces with I/O storage devices, as shown in FIG. 3.

According to an embodiment of the present invention, the host driver module 310 may be dynamically configured to accept communication requests for I/O device access from remote servers 340 and 350, via a SAN network interface card (NIC) 328 or other host adapters. Such a host driver module 310 (also known as "IOP" access module) can also be provided in a form of a hardware module, a combined hardware/software module, or a software module which can be easily downloaded (e.g., from a tangible medium such as a disk, or via the Internet) into the host operating system (OS) for allowing I/O storage devices 326 to operate independently from both the specific device controlled and the operating system (OS) of the host server 300.[8]

---

[7] The Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997 describes an open architecture for splitting a device driver into two modules in network system environment: (1) a host driver module – an upper module which is a host OS-specific portion of the driver system that interfaces with the host operating system (OS) installed at the host system; and (2) a device driver module – a lower module which is a device-specific portion of the driver system that interfaces with particular controllers and I/O storage devices; and. See pg 1-2 and 1-3.

[8] See page 8, lines 18-21 of Appellants' specification.

The device driver module 322, as shown in FIG. 3, may be included in an Input/Output Platform (IOP) 320 to provide an interface to I/O storage devices 326 and to control data transfer of I/O storage devices 326. The IOP 320 may be operatively connected to the host driver module 310, via a PCI bus 318, to control access to I/O storage devices 326, including transporting data from the host driver module 310 to the local IOP 320 and, vice versa.[9] In addition to the device driver module 322, the IOP 320 may also contain a communication layer 324 which defines an open, standard mechanism for communication between the host driver module 310 and the device driver module 322. The communication layer 324 may include a message layer which sets up a communication session, and a transport layer which defines how information will be shared. Collectively, the communication layer 324 may be responsible for managing all requests, and providing a set of Application Programming Interfaces (APIs) for delivering messages, along with a set of support routines that process the messages.[10]

The host driver module 310, as shown in FIG. 3, comprises a Local Transport 314 arranged to provide an interface to the IOP 320 supporting an array of I/O storage devices 326; a Remote Transport 316 arranged to provide an interface to another remote system such as remote

---

[9] An IOP 320 may also include a processor (not shown), a memory (not shown) that are managed independently from other processors within the host server, solely for processing I/O transactions.

[10] See page 9, lines 13-22 of Appellants' specification.

servers 340 and 350, via the SAN 330; and a Connection Manager 312 arranged to establish connection services and to create a direct call path between the Local Transport 314 and the Remote Transport 316 so as to provide direct access to I/O storage devices 326 without incurring the overhead of the operating system (OS) protocol stack and without coordinating special application-to-application messages.[11]

When the host driver module 310 is initialized, the Local Transport 314 scans the PCI bus 318 to locate and initialize all local IOPs 320 and builds an opaque "context" structure for each IOP found. The Remote Transport 316 prepares to accept requests from a remote server 340 or 350 through the SAN network interface card (NIC) 328. The Connection Manager 312 then queries the Local Transport 314 to determine the number of IOPs and builds a descriptor structure for each IOP. The IOP descriptor structure, as shown in FIG. 4, includes an exported table of function call pointers and the context required by the Local Transport 314 to communicate with each IOP. Next, the Connection Manager 312 establishes a SAN management communication channel through the Remote Transport 316, and waits for an external connection from a remote server 340 or 350 on a SAN 330.[12]

FIG. 4 illustrates an example IOP descriptor structure established during initialization for a

---

[11] See page 10, lines 12-21 of Appellants' specification.

[12] See page 11, lines 3-18 of Appellants' specification.

direct call interface between Local Transport 314 and Remote Transport 316 by the Connection Manager 312 of the host driver module 310 for an inbound message from a remote SAN server for direct access to local IOP 320 according to an embodiment of the present invention. The Local Transport 314 builds a list of different IOP context structures if different IOPs are available in the host server 300. Each IPO context structure includes different designations for direct access to each IOPs. In addition, the Local Transport 314 also has a send handler function which is a program interface to receive an inbound message from a remote SAN server for direct access to local IOP 320.[13]

The Connection Manager 312 builds an IOP descriptor structure for each IOP found. Each IOP descriptor structure includes an exported table of function call pointers such as IOP context pointer and send handler function pointer required by the Local Transport 314 to communicate with the IOP 320. The Remote Transport 316 builds an IOP connection structure including at least an IOP descriptor pointer which refers to the IOP descriptor structure of the Connection Manager 312 for making a direct call to the Local Transport 314 through the send handler function. In addition, the Remote Transport 316 also has a receive handler function which is a program interface to receive an inbound message from a remote server on a SAN for direct access to local IOP 320 and to deliver an outbound message to a remote server on a SAN. For an

---

[13] See page 11, lines 19-22 and page 12, lines 1-8 of Appellants' specification.

outbound message to a remote server on a SAN, data structures established for a direct call interface between Local Transport 314 and Remote Transport 316 by the Connection Manager 312 as shown in FIG. 4 remain the same. However, the communication directions between the send handler function in the Local Transport 314 and the receive handler function in the Remote Transport 316 are reversed.[14]

After initialization, a direct call interface between the Local Transport 314 and the Remote Transport 316 of the host driver module 310 is established to provide a direct, transparent access to I/O storage devices 326 within a system area network (SAN) cluster 330 that bypasses the traditional OS protocol stacks for efficient sharing of resources without incurring the overhead of the OS stack and coordinating special application-to-application messages between nodes of a SAN cluster 330.

First, in order to establish a service connection to an IOP 320, that is a logical connection between an IOP 320 attached to a host server 300 to a remote server 340 or 350, via a SAN 330, for the purpose of sending messages and transporting data there between, the remote server 340 or 350 in a SAN 330 exchanges messages with the Connection Manager 312 of the host drive module 310 based on a protocol specified by the network used and/or the Intelligent I/O Architecture. The Connection Manager 312 next advertises the presence of local IOPs that are available for external

---

[14] See page 12, lines 8-21 of Appellants' specification.

use.[15]

When a remote server 340 or 350 requests a service connection to an IOP 320 of the host server 300, the Connection Manager 312 passes the address of an IOP descriptor structure as shown in FIG. 4 to the Remote Transport 316 to establish the service connection. The IOP descriptor structure provides the Remote Transport 316 access to the function call pointers and context required by the Local Transport 314 to communicate directly with the IOP 320.

When a service connection to a local IOP 320 is established by the remote server 340 or 350 and the Connection Manager 312 of the host server 300, low latency and high-bandwidth messages passing between nodes is obtained by bypassing the layers of OS protocol stacks when sending and receiving messages. Data structure pointers are exchanged to establish a direct call relationship between software modules on the host server 300 within a SAN 330. To deliver an inbound message from a remote server 340 or 350 on a SAN 330, the receive handler function in the Remote Transport 314 simply refers to the IOP descriptor structure of the Connection Manager 312 to make a direct call to the send handler function in the Local Transport 314, providing the function with the IOP context and the message frame pointer. Likewise, an outbound message from the Local Transport 314 is delivered to a send handler function (not shown) in the Remote Transport 316. However, the outbound message includes a pointer to a structure containing the

---

[15] See page 13, lines 10-19 of Appellants' specification.

function address and context required by the Remote Transport 316 to send the message to a remote server 340 or 350 within a SAN 330.[16]

FIG. 5 illustrates an example step-by-step process of routing an inbound message or data from a remote server 340 or 350 of a SAN 300 through the Local Transport 314 and Remote Transport 316 by the Connection Manager 312 of a host driver module 310 according to an embodiment of the present invention. Each message may be transmitted in a series of frames. When a service connection is requested from a remote server, an inbound message frame from a remote server 340 or 350 on the SAN 330 is delivered to the receiver handler function in the Remote Transport 316 at step 510. The address of the IOP connection structure describing the service connection on which the inbound message was received (i.e., the context of the service connection) is located at the Remote Transport 316 at step 520. The IOP descriptor structure is located by using an IOP descriptor pointer to refer to the IOP descriptor structure of the Connection Manager 312.

Next, a context field in the inbound message frame is saved and replaced with a new context field for the Remote Transport 316 at step 530. The pointer to the IOP descriptor structure of the Connection Manager 312 is retrieved from the IOP connection structure at step 540. Then, the send handler function pointer and the IOP context pointer are read from the IOP descriptor

---

[16] See page 14, lines 3-16 of Appellants' specification.

structure of the Connection Manager 312 at step 550. The send handler function in the Local Transport 314 is called in order to pass the inbound message frame using the IOP context pointer in step 560.[17]

Likewise, FIG. 6 illustrates an example step-by-step process of routing an outbound message or data from a local I/O platform (IOP) 320 of a host server 300 to a remote server 340 or 350, via a SAN 330 according to an embodiment of the present invention. Again, each message is transmitted in a series of message frames. When an outbound message is routed to a remote server 340 or 350 on a SAN 330, an outbound message frame is delivered to the send handler function in the Local Transport 314 at step 610. The context field in the outbound message frame is read at step 620. This context is a pointer to a callback object structure in the Remote Transport 316 which contains the address of the Remote Transport IOP connection structure for a service connection.

Next, the callback function is called, passing the callback context pointer and the outbound message frame as outgoing parameters at step 630. At the Remote Transport 316, an outgoing message frame and the IOP connection structure address are delivered to the send handler function at step 640. A context field in the outgoing message frame is replaced with the saved context from the original inbound message frame as described with reference to FIG. 5, at step 650. Then, the

---

[17] See page 15, lines 1-12 of Appellants' specification.

outgoing message frame is sent out the service connection to a remote server 340 or 350 on a SAN 330 at step 660.[18]

As a result, the host driver system according to an embodiment of the present invention can export direct I/O storage device access without incurring the overhead of the traditional network infrastructure in the operating system (OS), and without using special application-to-application messages between end nodes within a SAN cluster.

VI. Issues

1. Whether claims 1-18 are unpatentable under 35 U.S.C. §103(a) as rendered obvious over Heil et al., U.S. Patent No. 6,173,374, as modified to incorporate selected features from the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997.

2. Whether claims 19-21 and 23-28 are unpatentable under 35 U.S.C. §103(a) as rendered obvious over Heil et al., U.S. Patent No. 6,173,374, as modified to incorporate selected features from the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola, U.S. Patent No. 6,321,279.

VII. Grouping of Claims

Independent claims 1, 7, 14 and 23 are argued separately, with the arguments necessarily carrying forward to their respective dependent claims 2-6, 8-13, 15-21 and 24-28. In addition,

---

[18] See page 16, lines 1-7 of Appellants' specification.

each of dependent claims 2-6, 8-13, 15-21 and 24-28 is also argued separately. Therefore, for purposes of the rejections under 35 U.S.C. §103(a), claims 1-21 and 23-28 stand or fall independently of each other under 37 C.F.R. §1.192(c)(5) for the reasons set forth in the arguments hereinbelow.

VIII. Arguments

1. Claims 1-18 are deemed patentable over the proposed combination of Heil et al., U.S. Patent No. 6,173,374 and the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997.

Claims 1-18 have finally been rejected under 35 U.S.C. §103(e) as being unpatentable over the Examiner's proposed combination of Heil et al., U.S. Patent No. 6,173,374 (hereinafter referred as Heil '374) and the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997. On pages 1-6 of the final Office action (Paper No. 12) dated on June 19, 2002, the Examiner asserts that Heil '374 and the Intelligent I/O (I_2O) Architecture Specification disclose or suggest all of the limitations of Appellants' claims 1-18 as pending on appeal.

For example, the Examiner asserts, on pages 1-3 of the final Office Action (Paper No. 12) dated on June 19, 2002, that Heil '374 as a primary reference discloses:

> a host driver configuration of a host driver module of a computer node on a server cluster, comprising: an input/output platform (IOP) installed in a host system arranged to control an array of local/remote (I/O) storage devices ... a host driver module installed in a

host system node (150 of Fig. 2) (col 10/line 28-50: I2O based partitions the device driver into one module of stackable drivers 210-280 containing an operating system and hardware code for accessing/controlling an array of storage devices, host device drivers within the host system node (150), col 10/lines 28-36, 51-58) of a computer node on a server cluster, (abstract: intelligent I_2O standard software, col. 10/lines 51-col 11/line 4), said device driver module includes software drivers (240, 250, 260 of Fig. 2) are arranged to control an array of local storage devices, driver software (240: I/O redirector) determines whether to satisfy a block I/O request locally or remotely, and coordinates the retrieval of data over a cluster with logically shared disks, clusters drives, comprising local stored data blocks, col 11/lines 5-11; and software drivers (ISM 250/HDM 260) control local disks (118), col 11/lines 12-27, said system driver module comprising:

a software driver (230 and 240 host interfaces of Fig. 2) (Local Transport) arranged to provide an interface to said input/output platform (lOP) (col 10/lines 57-58), and software driver (240) arranged to provide an interface to said local/remote IOP, col 10/lines 66-col 11/line 35.

software drivers (I/O shipping 270 and HDM 280 of Fig. 2) (Remote Transport) arranged to provide an interface to export local storage device access onto a computer network; a host driver module architecture on a server cluster for providing I/O access storage device data transfer between computer node(s) on a server cluster system (150) and another system (151); Heil teaches software drivers (270 and 280 of Fig. 2) is arranged to provide an interface layer to a computer network, which provides managing means to allow the host driver module (HBA) to determine whether to satisfy a block I/O request locally or remotely, managing means coordinates the retrieval of data over a cluster with logically shared disks, cluster drives, comprising local (118) or remote stored data blocks (123), col 11/lines 5-11; wherein software drivers (I/O shipping 270 and HDM 280 of Fig. 2) provide an interface layer for exporting local storage device onto a computer network, (I/O shipping 270) managing the reception of remotely generated requests from the other system network for local data to be exported onto the computer network, and (I/O shipping HDM 280) manages the communication medium (FC) transactions in support of shipping functions, col 11/lines 36-46.

software driver (I/O shipping HDM 280 of Fig. 2) (Connection Manager) arranged to establish connection services with remote servers on said computer network and coordinate functions responsible for creating a communication between the software driver

(250 of Fig. 2) and software driver (240 of Fig. 2) to provide access to the local or remote storage devices.

However Heil teachings of functions responsible for creating a communication path between the transport driver modules to provide access to the local storage devices are not denoted a "direct call path"; nor wherein that embedded intelligent I_2O standard software comprises an "IOP platform";

Specs teaches a message-based interfaces enable direct messages passing between any two device driver module for a particular class of I/O messages class, section 1.1.2.2, page 1-4, utility message (call) functions, table page 6-4 comprising message codes for accessing devices. Upon initialization the host then gives each IOP a list of all IOPs and the physical location of their inbound message queue. When an IOP wants to connect to another IOP, it sends the request to the respective IOP's inbound message queue location. The connection request and its reply convey information enabling the two IOPs to establish a direct path for exchanging messages, section 2.1.4.1, and page 2-11.

In accordance with I_2O standard architecture, this protocol comprises a single host entity and an intelligent I/O subsystem containing a number of I/O processors entities, host comprising application(s) and their resources, executing an operating system, Fig. 2-1, an IOP platform consists of a processor, memory and I/O devices, as per architecture's standard on page 2-1.

Therefore, Heil teachings wherein an host bus adapter (HBA) executing embedded intelligent software is arranged to control an array of local storage devices, inherently teaches that an input/output platform (IOP) arranged to control an array of local storage devices.

It would have been obvious to one ordinary skilled in the art at the time the invention was made to enable means for creating a direct call path between the driver modules to provide access to the local storage device, as taught by Specs, motivation would be enable a direct message passing between software driver layer for a particular class of I/O, such as those further specified by I_20 standards such as LAN ports, Ethernet or Token ring controllers, SCSI ports, etc providing an architecture that is operating-vendor-independent and adapts to existing system, creating scalable drives from high-end workstations to high-end server, as taught by Specs.". See page 2, Paper No. 5.

In other words, the Examiner alleges that Heil '374 discloses basically all features of Appellants' claims 1-18, except for "means for creating a direct call path between the driver modules to provide access to the local storage devices" and the use of an "IOP ... to control an array of local storage devices." The Examiner then cites section 1.1 - 2.2, pages 1-4 of the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997 which provides "the message-based interfaces enable direct message passing between any two device driver modules for a particular class of I/O (message class)" in order to support the conclusion that "[I]t would have been obvious ... to enable means for creating a direct call path between the driver modules to provide access to the local storage device" as allegedly suggested by the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997.

However, the Examiner's comments and assertions in support of the §103 rejection of Appellants' claims 1-18 are both factually and legally incorrect because neither Heil '374 nor the Intelligent I/O (I_2O) Architecture Specification, whether taken individually or in combination, discloses the claimed features as the Examiner has alleged. Virtually all the citations from either Heil '374 or the Intelligent I/O (I_2O) Architecture Specification are misplaced. Even more puzzling is the fact that no where in Heil '374 or the Intelligent I/O (I_2O) Architecture Specification is there disclosure of any module denoted as "Local Transport", "Remote Transport" and "Connection

Manager" as part of a host driver module - an upper module of a driver system as expressly identified in each of Appellants' independent claims 1, 7 and 14. Therefore the rejection of Appellants' claims 1-18 is respectfully traversed for reasons as discussed herein below.

A. The Examiner failed to establish a *prima facie* case of obviousness of Appellants' independent claims 1, 7, 14 because there is no factual evidence to support such a conclusion of obviousness.

In rejecting claims under 35 U.S.C. §103, the Examiner bears the initial burden of establishing a *prima facie* of obviousness. In re Rijckaert, 9 F.3d 1531, 1532, 26 USPQ2d 1955, 1956 (Fed. Cir. 1993); In re Oetker, 977 F.2d 1443, 1445 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). Only if this burden is met does the burden of coming forward with rebuttal argument or evidence shift to the Appellants. Rijckaert, 9 F.3d at 1532, 26 USPQ2d at 1956. When the references cited by the Examiner fail to establish a prima facie case of obviousness, the rejection is improper and must be overturned. In re Fine, 873 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). Obviousness under 35 U.S.C. §103 is a legal conclusion based on factual evidence, not a factual determination. Graham v. Deere, 383 U.S. 1, 148 USPQ 459. It is incumbent upon the Examiner to establish a factual basis to support the legal conclusion of obviousness. In re Fine, 873 F.2d at 1074, 5 USPQ2d at 1598. Determination of obviousness must be based on facts, and not on unsupported generalities. In re Warner, 379 F.2d 1011, 154

USPQ 173 (CCPA 1967); In re Freed, 425 F.2d 785, 165 USPQ 570 (CCPA 1970). Any deficiencies in the factual basis cannot be supplied by resorting to speculation or unsupported generalities. Id.

In the present situation, the Examiner has **not** met the initial burden of producing factual evidence to establish a *prima facie* case of obviousness. Each of Appellants' independent claims 1, 7, 14 and 22-23 defines a host driver module installed in a host system comprising a "Local Transport arranged to provide an interface to the IOP supporting an array of I/O storage devices"; a "Remote Transport arranged to provide an interface to another remote system, via the SAN"; and a "Connection Manager arranged to establish connection services and to create a direct call path between the Local Transport and the Remote Transport so as to provide direct access to I/O storage devices."

For example, independent claims 1 and 14 define a host driver module (IOP access module) installed in a host system (host server) for providing input/output device access between the host system and another system, via a data network, which comprises:

> a Local Transport arranged to provide an interface to an input/output platform (IOP) supporting an array of input/output devices;
>
> a Remote Transport arranged to provide an interface to said another system, via said data network; and
>
> a Connection Manager arranged to establish connection services and to create a direct call path between the Local Transport and the Remote Transport so as to provide

access to input/output devices.

Independent claim 7 further defines that such a host driver module (i.e., an upper module which is a host OS-specific portion of the driver system that interfaces with a host operating system) is part of an overall driver module which also contains a device driver module arranged to provide an interface to the array of local storage devices (i.e., a lower module which is a device-specific portion of the driver system that interfaces with I/O devices).

As expressly defined in each of Appellants' independent claims 1, 7 and 14, the host driver module 310 as installed in a host system (as part of a host operating system) to access an input/output platform (IOP) includes a Connection Manager 312, a Local Transport 314 and a Remote Transport 316 for the purposes of exporting device access to remote devices on a data network (SAN), as shown in FIG. 3. For example, Local Transport 314 is arranged to provide an interface to IOP 320, via the PCI bus 318 supporting an array of IO devices 326. Remote Transport 316 is arranged to provide an interface to remote devices (remote servers) via a SAN 330. Connection Manager 312 is utilized to establish connection services and create a direct call path between the Local Transport 314 and the Remote Transport 316 so as to provide access to IO devices 326. Data structure pointers as shown in FIG. 4 are then exchanged between the Local Transport 314 and the Remote Transport 316 by way of the Connection Manager 312 in order to establish a direct-call relationship between the Local Transport 314 and the Remote Transport 316

in order to avoid incurring the overhead of the operating system (OS) protocol stack and coordinating special application-to-application messages as required by the SAN cluster.

In contrast to Appellants' independent claims 1, 7 and 14, Heil '374 as a primary reference discloses a technique for I/O shipping of requests using peer-to-peer communications between host bus adapters ("HBAs") installed in a respective node in a data network as shown in FIG. 1. Specifically, Heil '374 discloses the use of one or more intelligent host bus adapters "HBA" (for example, a single HBA 117 shown in FIG. 2, or alternatively, two HBAs 180, 181 shown in FIG. 5A installed to reduce the workload within each node) installed in a host system (host node) of a clustered data network for processing IO requests received from the host system. Each intelligent HBA also serves as a network interface card (NIC) connected to a peer HBA via a Fibre Channel backbone 121 (high-speed communication medium) and contains therein a directory within memory 116 for storing location information regarding blocks of data stored in I/O storage devices and software for searching the directory to determine whether to locally or remotely retrieve blocks of data. Independent of the host system, the intelligent HBA installed in such a host system distributes I/O requests to the appropriate HBA (installed in another node in a clustered data network) in response to the directory search. Such an intelligent HBA is operable to establish and maintain communications with at least one other HBA installed in another node of the clustered data network. For example, on column 4, lines 51-67, Heil '374 describes that,

> "an HBA receives a block I/O request from the host and searches through the directory in HBA memory to determine whether the block can be found locally or remotely. If the search through the directory does not find a particular data block, the HBA may poll (e.g., query) peer HBAs [installed in different nodes in a clustered data network] to determine which HBA can satisfy a particular block I/O request.
>
> If blocks are available remotely, then the initiating HBA establishes communications with its peer HBA residing in the node containing the requested data blocks (e.g., peer-to-peer communications among HBAs). The I/O request is 'shipped' to the peer HBA, which performs the requisite processing. Data returned from the HBA performing the remote processing (i.e., a read request) is passed by the initiating HBA to the requesting host. Otherwise, the initiating HBA retrieves the requested data blocks from the local disks."

According to Heil '374, "implementing I/O shipping within the HBA virtually eliminates overhead resulting from processing I/O requests within the host systems previously imposed when sharing storage resources. That is, the host [OS] software no longer has to handle the I/O shipping function, and inter-processor communication networks no longer need to carry I/O traffic in addition to communication traffic." See column 5, lines 18-24 of Heil '374.

In other words, Heil '374 discloses nothing more than an especially designed HBA which is equivalent to Appellants' disclosed "network interface card" (NIC) 328 as shown in FIG. 3, albeit there is new functionality and added intelligence that is not commonly available for off-the-shelf host adapters or host interface cards used in connection with Appellants' disclosed invention. Such a HBA 117 of Heil '347 does **not** constitute a host driver module (IOP access module) as defined in Appellants' independent claims 1, 7, 14 and 22-23, and shown in FIG. 3 as a separate module

from the NIC 328. In fact, Heil '374 does **not** disclose or suggest the use of any host driver module installed in a host system to provide access to I/O devices as falsely and repeatedly alleged by the Examiner.

As shown in FIGs. 1-2 of Heil '374, the intelligent HBA 117 or HBAs 180 and 181 (shown in FIG. 5A) are hardware adapters arranged below the PCI bus 116.5 to connect to a clustered data network or local I/O devices, as is customary by any other host adapter or Appellants' disclosed "network interface card 'NIC'". Such intelligent HBAs are **not** part of any host operating system (OS) as mistakenly asserted by the Examiner. Since the intelligent HBAs as described by Heil '374 are hardware adapters arranged well below the PCI bus 116. 5, as shown in FIGs. 1-2, and are **not** part of any host operating system (OS), such intelligent HBAs do **not** and cannot be interpreted to constitute a host driver module as defined by Appellants' independent claims 1, 7 and 14. Likewise, since the intelligent HBAs of Heil '374 do **not** constitute and cannot be interpreted to constitute Appellants' claimed "host driver module", such intelligent HBAs do **not** and cannot be interpreted to include "a Local Transport arranged to provide an interface to an input/output platform (IOP) supporting an array of input/output devices;" "a Remote Transport arranged to provide an interface to said another system;" and "a Connection Manager arranged to establish connection services and to create a direct call path between the Local Transport and the Remote Transport so as to provide access to IO devices" as generally defined in Appellants'

independent claims 1, 7 and 14.

Nevertheless, the Examiner has selected different elements, sometimes same elements, contained in the intelligent HBA of Heil '374 to read on each claimed element of Appellants' claimed "host driver module".

For example, the Examiner asserts that "a software driver (230 and 240 host interfaces of Fig. 2)" [referring to column 10, lines 57-58] of Heil '374 constitutes, or at least can be interpreted to constitute, Appellants' claimed "Local Transport arranged to provide an interface to an input/output platform (IOP) supporting an array of input/output devices" as expressly defined in Appellants' independent claims 1, 7 and 14.

This assertion is fundamentally flawed for several reasons. First of all, FIG. 2 of Heil '374 is a block diagram of software layers of both the host CPU 100 and the intelligent HBA 117 shown in FIG. 1. As shown in FIG. 2 and described from column 10, line 51 extending to column 11, line 46 of Heil '374, the software layers of the host CPU 100 include host layers 200 (i.e, operating system "OS" layers), and a host driver 210 arranged above the PCI bus 220. In contrast to the software layers of the host CPU 100, the software layers of the intelligent HBA 117 are below the PCI bus 220, including a host interface layer 230; an I/O redirector software 240; device driver modules (DDM) such as local RAID, cache intermediate service modules (ISMs) 250 and local

storage hardware device modules (HDMs) 260[19]; and I/O shipping modules 270 and 280. Since the host interface layer 230 and the I/O redirector software 240, as shown in FIG. 2, are software layers of the intelligent HBA 117, and are **not** part of the host driver module [see above the PCI bus 220], neither the host interface layer 230 nor the I/O redirector software 240, alone or in combination, can be reasonably interpreted to constitute Appellants' claimed "Local Transport" in a host driver module that is "arranged to provide an interface to an input/output platform (IOP) supporting an array of input/output devices" as expressly defined in Appellants' independent claims 1, 7 and 14. Rather, the host interface layer 230 and the I/O redirector software 240 of Heil '374 along with other hardware device modules (HDMs) 250-280 are software layers of the intelligent HBA 117 and, as a result, can be interpreted as part of the input/output platform (IOP).

Secondly, as shown in FIG. 2 and described extensively on column 10, lines 57-59 and column 11, lines 3-8 of Heil '374, the host interface layer 230 is simply used to manage communication between the host and the HBA's embedded intelligence. Likewise, the I/O redirector software 240 is simply used to make the decision whether to satisfy a block I/O request locally or remotely, i.e., to search an internal directory to determine whether a block I/O request can be found locally or remotely. Neither the host interface layer 230 nor the I/O redirector

---

[19] Hardware Device Modules (HDMs) and Intermediate Service Modules (ISMs) are often referred to collectively as "Device Driver Modules" (DDMs).

software 240, whether taken individually or in combination, can be reasonably interpreted to constitute Appellants' claimed "Local Transport arranged to provide an interface to an input/output platform (IOP) supporting an array of input/output devices" as expressly defined in Appellants' independent claims 1, 7 and 14.

The Examiner also asserts that "software drivers (I/O shipping 270 and HDM 280 of Fig. 2)" of Heil '374 constitute, or at least can be interpreted to constitute, Appellants' claimed "Remote Transport arranged to provide an interface to another system, via a data network" as expressly defined in Appellants' independent claims 1, 7 and 14.

Again, this assertion is also flawed for at least two reasons. First, the I/O shipping modules 270 and 280, as shown in FIG. 2 and described on column 11, lines 36-43 of Heil '374, are software layers of the intelligent HBA 117, and are **not** part of the host driver module [see above the PCI bus 220] as expressly identified by Appellants' independent claims 1, 7 and 14. Secondly, such I/O shipping modules 270 and 280 are used to ship out I/O requests, via a data network 152, for directory updates, updating the directory, managing the transmission of locally generated requests for remote data out of the data network 152, and managing the reception of remotely generated requests from the data network 152 for local data. As a result, such I/O shipping modules 270 and 280 cannot be reasonably interpreted to constitute Appellants' claimed "Remote Transport arranged to provide an interface to another system, via a data network" as expressly

defined in Appellants' independent claims 1, 7 and 14.

The Examiner further asserts that "software driver (I/O shipping HDM 280 of Fig. 2)" of Heil '374 constitutes, or at least can be interpreted to constitute, Appellants' claimed "Connection Manager arranged to establish connection services and to create a direct call path between the Local Transport and the Remote Transport so as to provide access to IO devices" as expressly defined in Appellants' independent claims 1, 7 and 14.

Likewise, this assertion is also flawed for several reasons. First of all and most obviously, if the I/O shipping HDM 280, as shown in FIG. 2 of Heil '374, has been cited to read on Appellants' claimed "Remote Transport", the same I/O shipping HDM 280 cannot now be used to read on another claimed element, i.e., Appellants' claimed "Connection Manager". For the Examiner to do so can only demonstrate the arbitrary and capricious nature of the outstanding rejection. Secondly and as previously discussed, the I/O shipping HDM 280 is a software layer of the intelligent HBA 117, and is **not** part of a host driver module as expressly identified by Appellants' independent claims 1, 7 and 14. Lastly, the I/O shipping HDM 280, as previously discussed, is only used to ship out I/O requests, via a data network 152, for directory updates, updating the directory, managing the transmission of locally generated requests for remote data out of the data network 152, and managing the reception of remotely generated requests from the data network 152 for local data. See column 11, lines 36-43 of Heil '374. As a result, such an I/O shipping HDM 280

does **not** constitute and cannot be interpreted to constitute, Appellants' claimed "Connection Manager arranged to establish connection services and to create a direct call path between the Local Transport and the Remote Transport so as to provide access to IO devices" as expressly defined in Appellants' independent claims 1, 7 and 1According to an embodiment of the present invention, the creation of a direct call path between the Local Transport and the Remote Transport is essential in terms of providing direct access to I/O storage devices without incurring the overhead of the operating system (OS) protocol stack and without coordinating special application-to-application messages.[20]

In fact, there is **no** element anywhere in Heil '374 that can be interpreted to read on Appellants' claimed "Connection Manager arranged ... to create a direct call path between the Local Transport and the Remote Transport so as to provide access to I/O devices." According to an embodiment of the present invention, the creation of a direct call path between the Local Transport and the Remote Transport is essential in terms of providing direct access to I/O storage devices without incurring the overhead of the operating system (OS) protocol stack and without coordinating special application-to-application messages.[21] As a result, there is no need for the "Connection Manager" to search any type of directory to determine if a block I/O request can be

---

[20] See page 10, lines 12-21 of Appellants' specification.

[21] See page 10, lines 12-21 of Appellants' specification.

found locally or remotely or otherwise, perform any other undesirable functions required by Heil '374.

As a secondary reference, the Intelligent I/O (I_2O) Architecture Specification has been cited by the Examiner for disclosing "message-based interfaces [which] enable direct message passing between any two device driver modules for a particular class of I/O (message class). See section 1.1. However, Appellants remain puzzled as to how this disclosure from the Intelligent I/O (I_2O) Architecture Specification provides suggestion or motivation to one skilled in the art to "enable means for creating a direct call path between the driver modules to provide access to the local storage devices".

Section 1.1 of the Intelligent I/O (I_2O) Architecture Specification refers to the ability of "message-based interfaces [to] enable direct message passing between any two device driver modules for a particular class of I/O (message class)". In a typical I/O subsystem or I/O platform (IOP), there are multiple device driver modules, and each of these device driver modules is designed for a specific I/O device, such as, for example, LAN ports (e.g., Ethernet or Token Ring controllers); random block storage devices (e.g., hard disk drives and CD-ROM drives); sequential storage devices and variable block-size drives (e.g., tape drives); SCSI ports; SCSI devices; IDE controllers and devices; flopping disk devices etc. Therefore, if such an IOP contains any two device driver modules for a particular class, i.e., specific I/O device, direct messages may be

passed between these two device driver modules using the message-based interfaces as described in accordance with the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997.

Again, the two device driver modules as specified by the Intelligent I/O (I_2O) Architecture Specification are exemplary of Appellants' claimed "device driver module" 322 - a lower module [see below the PCI 318] which is a device-specific portion installed in the IOP 320 that interfaces with I/O storage devices, as shown in FIG. 3. These device driver modules as referred to by the Examiner are **not** and cannot be interpreted to read on specific modules of Appellants' claimed "host driver module 310" - an upper module [see above the PCI 318] which is a host OS-specific portion installed as part of an operating system (OS) of the host server 300, that is separate and distinct from the device driver module 322 - a lower module [see below the PCI 318] which is a device-specific portion that interfaces with I/O storage devices, as shown in FIG. 3. In fact, such a direct message passing procedure is implicit in Appellants' claimed "device driver module 322" since Appellants' claimed "overall driver system" is designed specifically in accordance with the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997. See page 10, lines 18-21 of Appellants' specification.

Since the direct message passing procedure as described in Section 1.1 of the Intelligent I/O (I_2O) Architecture Specification is only applicable in Appellants' claimed "device driver module 322", and is **not** applicable in Appellants' claimed "host driver module" 310, such a direct message

passing procedure has absolutely **no** bearing as to how Appellants' claimed "Connection Manager" [or any means] is able to create a direct call path between the Local Transport and the Remote Transport so as to provide access to I/O devices," nor anything specific modules of Appellants' claimed "host driver module" 310.

Accordingly, in view of the foregoing reasons and the complete failure of Heil '374 and the cited portion of the Intelligent I/O (I_2O) Architecture Specification to disclose the claimed features as alleged by the Examiner, Appellants respectfully request that the rejection of Appellants' independent claims 1, 7 and 14 be reversed.

B. The Examiner erred in making the modification of Heil '374 to incorporate selected features from the Intelligent I/O (I_2O) Architecture Specification to arrive at Appellants' independent claims 1, 7 and 14 because there is no suggestion in either Heil '374 or the Intelligent I/O (I_2O) Architecture Specification to support such a modification.

> The mere fact that a prior art device could be modified to produce the claimed invention does not justified an obviousness rejection unless the prior art suggested the desirability of the modification (emphasis in original).

In re Laskowski, 871 F.2d 115, 10 USPQ2d 1397 (Fed. Cir. 1999); quoting In re Gordon, 733 F.2d 900, 902, 221 USPQ 1125, 1127 (Fed. Cir. 1984). In re Fine, 873 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). In re Freed, 425 F.2d 785, 165 USPQ 570 (CCPA

1970). Therefore, even assuming *arguendo* that the cited section of the Intelligent I/O (I_2O) Architecture Specification actually discloses or suggests, which Appellants do not believe, that some sort of "direct call path between the driver modules" can be created "to provide access to the local storage devices, there is still **no** reason for an artisan to make the modification in the manner suggested by the Examiner in order to arrive at Appellants' independent claims 1, 7 and 14. This is particular true when neither Heil '374 nor any section of the Intelligent I/O (I_2O) Architecture Specification discloses Appellants' claimed "host driver module" installed in a host system comprising a "Local Transport arranged to provide an interface to the IOP supporting an array of I/O storage devices"; a "Remote Transport arranged to provide an interface to another remote system, via the SAN"; and a "Connection Manager arranged to establish connection services and to create a direct call path between the Local Transport and the Remote Transport so as to provide direct access to I/O storage devices" as expressly defined in each of Appellants' independent claims 1, 7 and 14.

In view of the complete failure of Heil '374 and the Intelligent I/O (I_2O) Architecture Specification to suggest the modification, Appellants respectfully submit that the proposed combination of Heil '374 and the Intelligent I/O (I_2O) Architecture Specification does not make obvious Appellants' claimed invention. Accordingly, the rejection of Appellants' independent claims 1, 7 and 14 should be reversed.

C. Dependent claims 2-6, 8-13 and 15-18 are deemed separately patentable over Heil '374, as modified to incorporate the Intelligent I/O (I_2O) Architecture Specification.

Claims 2-6, 8-13 and 15-18 which depend from claims 1, 7 and 14 are deemed patentable from claims 1, 7 and 14 if their parent claims 1, 7 and 14 are patentable. Hartness Int'l, Inc., v. Simplicatic Eng'g Co., 891 F.2d 1100, 1108, 2 USPQ2d 1826, 1831 (Fed. Cir. 1987); In re Abele, 684 F.2d 909, 214 USPQ 682, 689 (CCPA 1982) *see also* In re Sernaker, 702 F.2d 989, 991, 217 USPQ 1, 3 (Fed. Cir. 1983).

Even assuming *arguendo* that independent claims 1, 7 and 14 are not patentable under 35 U.S.C. §103(a), which Appellants do not believe, dependent claims 2-6, 8-13 and 15-18 are separately patentable from parent claims 1, 7 and 14 for reasons presented herein below.

For example, dependent claims 4 and 15 further define that the IOP comprises: "one or more IO processors, IO devices, a device driver module and a communication layer which defines a mechanism for communications between the Local Transport and the device driver module". These features are **not** disclosed anywhere in Heil '374 or the Intelligent I/O (I_2O) Architecture Specification.

However, the Examiner asserts that the combined teachings of Heil '374 and the Intelligent I/O (I_2O) Architecture Specification further disclose,

> "wherein said IOP which comprises: at least one or more input/output processors (see Specs, page 2-1); at least one storage device as said input/output devices (Heil: disks 118, col 11/lines 7-35); a device driver module arranged to control access via interface means with said storage device (Heil: software driver 260, col 11/lines 7-12, 28-35); a communication layer which defines a mechanism for communications between the ISM driver 250 (Local Transport) and the HDM (260); (Heil: software driver 250, col 11/lines 12-27, supporting communication between 250 and 260, see Specs messaging (communication) layer section 2.1.3, page 2-4, Fig. 2-3." See Page 4 of Paper No. 12.

This assertion is incorrect. Again, Heil '374 does **not** disclose the specifics of Appellants' claimed "host driver module" - an upper [host OS-specific] module that interfaces a host operating system (OS) that is separate and distinct from "a device driver module"- a lower [device-specific] module that interfaces I/O devices. Rather, Heil '374 only discloses the specifics of interface layers, such as a host interface layer 230 and an I/O redirector software 240, and device driver modules such as ISMs and HDMs 250-280 of an intelligent HBA 117 or some other forms of input/output platforms (IOPs) as suggested by the Intelligent I/O (I_2O) Architecture Specification. However, neither Heil '374 nor the Intelligent I/O (I_2O) Architecture Specification discloses the use of a communication layer installed in an IOP which defines a mechanism for communications between the Local Transport and the device driver module as expressly defined in dependent claims 4 and 15.

Dependent claims 5, 9 and 16 further define that the "communication layer is responsible for managing all service requests and providing a set of Application Programming Interfaces (APIs) for

delivering messages, along with a set of support routines that process the messages". Dependent claims 6 and 10 further define that the "communication layer comprises a message layer which sets up a communication session, and a transport layer which defines how information will be shared". Again, none of these features is disclosed or suggested by Heil '374 or the Intelligent I/O (I_2O) Architecture Specification.

Nevertheless, the Examiner asserts that the combined teachings of Heil '374 and the Intelligent I/O (I_2O) Architecture Specification further disclose,

> "wherein said communication layer is responsible for managing and dispatching all service requests (see Specs: section 2.1.3, page 2-4, Fig. 2-3 communication layer is a network of object instance) and providing a set of APIs for delivering messages along with a set of support routines that process the message (see Specs: message is an interface between the modules, this interface is a set of APIs that provide transport and message services, page 2-5), and is comprised of a message layer which sets up a communication session (see Specs: section 2.1.6, page 2.18), and transport layer which defines how information will be shared (Heil coordinate retrieval of shared information, col 10/lines 66-col 11/line 11)." See pages 4-5, Paper No. 15.

Again the assertion is incorrect. None of the cited portions of Heil '374 and the Intelligent I/O (I_2O) Architecture Specification discloses what the Examiner has alleged. Heil '374 only discloses the specifics of device driver modules installed in an intelligent HBA, and does **not** disclose the specifics of Appellants' claimed "host driver module" - an upper [host OS-specific] module that interfaces a host operating system (OS) that is separate and distinct from "device driver

modules"- lower [device-specific] modules that interface I/O devices. If Heil '374 does **not** disclose any "host driver module" and any "IOP" as defined in claims 4 and 15, then there is **no** possibility of any disclosure from Heil '374 and the Intelligent I/O (I_2O) Architecture Specification of any claimed "communication layer [of an IOP] for managing all service requests and providing a set of Application Programming Interfaces (APIs) for delivering messages, along with a set of support routines that process the messages" as defined in claims 5, 9 and 16, and any claimed "communication layer comprises a message layer which sets up a communication session, and a transport layer which defines how information will be shared" as defined in dependent claims 6 and 10.

Dependent claim 11 further defines that "the host driver module and the device driver module constitute a single device that is portable across a plurality of operating systems and host network platforms, and works interoperably with a plurality of storage devices and operating systems". Again, this feature is **not** disclosed anywhere in Heil '374 or the Intelligent I/O (I_2O) Architecture Specification.

Nevertheless, the Examiner asserts that the combined teachings of Heil '374 and the Intelligent I/O (I_2O) Architecture Specification further disclose,

> "wherein said system driver module and said device driver module constitute a single device (Heil: col 4/lines 43-47, stackable device drivers in a single module) that is portable across a plurality of operating systems and host network platforms (see Specs: to enable device drivers to port across target processors, device driver

source code is written in ANSI C, section 1.2, page 1-4), and works interoperably with a plurality of storage devices and operating systems. See page 5, Paper No. 12.

Again, this assertion is also incorrect. None of the cited portions of Heil '374 and the Intelligent I/O (I_2O) Architecture Specification discloses what the Examiner has alleged. This is because Heil '374 simply describes stackable "device driver modules" installed an intelligent HBA or some other forms of IOPs as suggested by the Intelligent I/O (I_2O) Architecture Specification. Heil '374 does **not** disclose the specifics of Appellants' claimed "host driver module" - an upper [host OS-specific] module that interfaces a host operating system (OS) that is separate and distinct from "device driver modules"- lower [device-specific] modules that interface I/O devices. Since Heil '374 does **not** disclose Appellants' claimed "host driver module", there is **no** possibility of any incorporation of both the "host driver module" and "device driver module" as defined in claim 11.

In view of the foregoing explanations, and in view of the fact that neither Heil '374 nor the Intelligent I/O (I_2O) Architecture Specification, whether taken in combination or individually, discloses and suggests Appellants' dependent claims 2-6, 8-13 and 15-17, Appellants respectfully request that the rejection of dependent claims 2-6, 8-13 and 15-17 be reversed as well.

2. Claims 19-21 and 23-28 are deemed patentable over the proposed combination of Heil et al., U.S. Patent No. 6,173,374, the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola, U.S. Patent No. 6,321,279.

Claims 19-28 have finally been rejected under 35 U.S.C. §103(e) as being unpatentable over the Examiner's proposed combination of Heil et al., U.S. Patent No. 6,173,374 (hereinafter referred as Heil '374), the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola, U.S. Patent No. 6,321,279. As previously indicated, independent claim 22 has been canceled without prejudice or disclaimer. In addition, claims 19-21 are dependent claims which depend upon previously indicated independent claim 14 and an intervening claim 15. As a result, claims 19-21 should, and will be addressed separately from claims 23-28 herein below.

Dependent claim 19 which incorporates the specifics of Appellants' claimed "host driver module" as including "a Local Transport", "a Remote Transport" and "a Connection Manager" that are separate and distinct from a device driver module - a lower [device-specific] module installed in an IOP that interfaces I/O devices, further defines,

> "wherein, upon initialization, said Local Transport scans the local bus so as to locate and initialize all local input/output platforms (IOPs) and builds an opaque "context" structure for each input/output platform (IOP), wherein said Remote Transport prepares to accept requests from a remote server through said computer network, and wherein said Connection Manager queries said Local Transport so as to determine the number of input/output platforms (IOPs), builds an IOP descriptor structure for each input/output platform (IOP) which includes an exported table of function call pointers and the context required by the Local Transport to communicate with the input/output platform (IOP), and finally establishes a network management communication channel through the Remote Transport, which waits for an external connection from said remote server on said computer network for exporting local device access onto said computer network using said direct call path between the Local Transport and the Remote Transport."

In other words, the Local Transport 314 scans the PCI bus 318 to locate and initialize all local IOPs and builds an opaque "context" structure for each IOP found. The Remote Transport 316 then prepares to accept requests through network interface card (NIC) 328. The Connection Manager 312 then queries the Local Transport 314 to determine the number of IOPs, builds a descriptor structure for each IOP which includes an exported table of function call pointers and the context required by the Local Transport 314 to communicate with the IOP, and establishes a management communication channel through the Remote Transport 316, which waits for an external connection from a remote server via a SAN for exporting local device access using a direct call path between the Local Transport 314 and the Remote Transport 316.

The host driver module 310 allows, for example, a distributed database running on a cluster of servers to share and directly access all the storage in the cluster transparently. As a result, the overhead incurred by the OS stack on the remote node is avoided via "short-circuiting" at the driver level, and no unique database is required to generate special application-to-application messages to remote nodes in order to access IO storage devices located on remote storage.

Nevertheless, the Examiner asserts, in support of the rejection of dependent claim 19, that,

> Specs: teach building a list of designations for each IOP, wherein after the IOP loads, it executes its initialization sequence, causing the IOP to scan its physical adapters, load and initialize appropriate devices, and build a logical configuration table, section 2.1.7.2, page 2.25, located IOP is added to the system configuration table, host then gives each IOP a list of all IOPs, section 2.1.4.1, page 2-11, said table lists all the IOPs registered devices and their availability, item 8, page 4-65, for each device registered, the

driver provides a list of message handlers, one for each message function, received messages handlers are used to builds a message dispatch table, section 2.2.4.3, page 2-32, each registered I_2O device provides message handlers that process the messages to the device, the identify of the event handler is accompanied by a context variable so that the message variable can determine the associated I/O transaction of each registered IOP, section 2.2.4.1, section Page 2-32.

wherein software driver 240 (Remote Transport) prepares to accept requests from a remote server through said computer network, (Heil: Software drivers 240 arranged to manage the reception of remotely generated requests from the network for local data to be exported onto the computer network, col 11/lines 36-46, remote computer nodes of a server cluster, abstract), and

means for building an IOP descriptor structure for each input/output platform (IOP) which includes an exported table of function call pointers and the context required by the software driver 250 to communicate with the input/output platform (IOP);

Specs: Each I_2O device is created with a unique TID and associated with an event queue, IOP and driver communicate via APT function calls. Driver supplies a pointer to the (IOP descriptor function table) message dispatch table when the device is created. This table lists Function codes, their priorities and message handlers for processing request messages. When communicating with the IOP, the IOP receives a request message to that TID, the IOP uses the message's Function code (function call), queues the request to the event queue. If the driver sends requests, it must register their reply handlers (i.e. send/receive handlers) and priorities via an API function call. Handles are placed in a structure and which returns an InitiatorContext value identifying that structure. When a Driver sends a request, it uses the appropriate InitiatorContext value. When a reply is received, the IRTOS retrieves the reply handler and priority from a structure identified by the InitiatorContext field and then queues the event, section 5.1.5, page 5-4, section 5.2.6, page 5-14; In this manner an IOP descriptor structure for each input/output platform (IOP) is built, which includes an exported table of function call pointers and the context required by the driver to communicate with the input/output platform (IOP); connection using said direct call path between the driver 250 and driver 240, discussed above in claims 1 and 14;

and means for establishing a network management communication channel through the driver 240 (Heil, col 11/lines 54-65, shipping via 240, col 10/line 65-col 10/line 17),

and means which waits for a connection (see Specs, table 4-5, page 4-16, connection_fail, timeout waiting for response), however neither Heil nor Specs teach determining the number of IOP;

Bonola teaches means for determining upon initialization and a starting process, the number of CPU's present in the computer system along with the context of the computer system, col 9/lines 55-58, teaching means for querying so as to determine the number of input/output processing (IOP), col 8/line 5-13, 58-64);

It would have been obvious to one ordinary skilled in the art at the time the invention was made to modify existing teachings with means for determining the number of IOP as taught by Bonola motivation would be to prevent error in the installation phase determining the actual of detected processor, utilizing said number and the context information to initialize the drivers, supporting the allocation of shared memory without requiring extra memory, as in the prior art.

See Pages 6-7 of Paper No. 12.

However, no where in Heil '347, the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola '279, is there disclosure of any host driver module including modules denoted as "Local Transport", "Remote Transport" and "Connection Manager" designated to perform the functions as identified by the Examiner. In fact, all cited portions of Heil '347, the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola '279 are misplaced.

Recognizing the inherent deficiencies of Heil '347, the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola '279, the Examiner has expressly conceded patentability of Appellants' claim 19. Specifically, on the Examiner Interview Summary attached to

the final Office Action (Paper No. 12) dated on June 19, 2002, the Examiner has expressly indicated if the limitation,

> " wherein said Connection Manager queries said Local Transport so as to determine the number of input/output platforms (IOPs), builds an IOP descriptor structure for each input/output platform (IOP) which includes an exported table of function call pointers and the context required by the Local Transport to communicate with the input/output platform (IOP)"

of dependent claim 19 is incorporated into each of Appellants' independent claims 1, 7, 12, 19 and 22, all claims 1-28 as pending on this Appeal will be allowed and the instant application will be in condition for issuance.[22] However, Appellants believe such an incorporation is entirely unnecessary since each of Appellants' independent claims 1, 7, 12, 19 and 22 is already deemed patentably distinguishable over the Examiner's proposed combination of Heil '374, the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola '279 for reasons discussed in this Appeal Brief.

For reasons discussed and in view of the Examiner's express admission that claim 19 is allowable over the proposed combination of Heil '374, the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola '279, Appellants respectfully request that the rejection of claim 19 and its dependent claims 20-21 be reversed.

---

[22] Also see Advisory Action (Paper No. 14) dated on July 12, 2002 and Advisory Action (Paper No. 15) dated on July 31, 2002.

Likewise, independent claim 23 defines a process of establishing a service connection to a local IOP connected to a local bus using a system driver module in response to a request from a remote server on a system area network (SAN). Such a driver module may be initialized and activated as follows:

> beginning initialization of said driver module which provides access to a local storage system while bypassing protocol stacks of a host operating system, said system driver module comprising a Local Transport which provides direct access to the local storage device system, a Remote Transport which interfaces to other nodes of said system area network, and a Connection Manager which provides connection services and coordinates functions responsible for creating a direct call path between the Local Transport and the Remote Transport;
>
> scanning, at said Local Transport, the local bus to locate and initialize all local input/output platforms (IOPs), and building an IOP context structure for each input/output platform (IOP) found;
>
> preparing, at said Remote Transport, to accept a request for a service connection from said remote server on said system area network;
>
> asking, at said Connection Manager, whether said Local Transport determines the number of input/output platforms (IOPs), and building a descriptor structure for each input/output platform (IOP) which includes an exported table of function call pointers and the context required by the Local Transport to communicate with the input/output platform (IOP); and
>
> establishing a system area network management communication channel through the Remote Transport, which waits for an external connection from said remote server on said system area network for exporting local device access onto said system area network using said direct call path between the Local Transport and the Remote Transport.

Again, the driver module is defined as including a "Local Transport", a "Remote Transport"

and a "Connection Manager". In addition, the Local Transport 314 scans the PCI bus 318 to locate and initialize all local IOPs and builds an opaque "context" structure for each IOP found. The Remote Transport 316 then prepares to accept requests through network interface card (NIC) 328. The Connection Manager 312 then queries the Local Transport 314 to determine the number of IOPs, builds a descriptor structure for each IOP which includes an exported table of function call pointers and the context required by the Local Transport 314 to communicate with the IOP, and establishes a management communication channel through the Remote Transport 316, which waits for an external connection from a remote server via a SAN for exporting local device access using said direct call path between the Local Transport 314 and the Remote Transport 316. As a result, the overhead incurred by the OS stack on the remote node is avoided via "short-circuiting" at the driver level, and no unique database is required to generate special application-to-application messages to remote nodes in order to access IO storage devices located on remote storage.

Again, independent claim 23 contains limitations that are similar to that of dependent claim 19, and for reasons discussed previously and in view of the Examiner's express admission that claim 19 is allowable over the proposed combination of Heil '374, the Intelligent I/O (I_2O) Architecture Specification, Version 1.5, March 1997, and Bonola '279, Appellants respectfully request that the rejection of independent claim 23 and its dependent claims 24-28 be reversed.

Even assuming *arguendo* that independent claim 23 is not patentable under 35 U.S.C.

§103(a), which Appellants do not believe, dependent claims 24-28 are separately patentable from parent claim 23 for reasons presented herein below.

For example, dependent claim 24 further defines that the IOP comprises: "a device driver module which interfaces the local storage devices, and which controls an array of local storage devices; and a communication layer which defines a mechanism for communications between the system [host] driver module and the device driver module". These feature are **not** disclosed anywhere in Heil '374, the Intelligent I/O (I_2O) Architecture Specification, or Bonola '279.

Dependent claim 25 further defines that the "communication layer is responsible for managing all service requests and providing a set of Application Programming Interfaces (APIs) for delivering messages, along with a set of support routines that process the messages, and is comprised of a message layer which sets up a communication session, and a transport layer which defines how information will be shared". Again, none of these features is disclosed or suggested by Heil '374, the Intelligent I/O (I_2O) Architecture Specification, or Bonola '279.

Dependent claim 26 further defines that "the host driver module and the device driver module constitute a single device that is portable across a plurality of operating systems and host network platforms, and works interoperably with a plurality of storage devices and operating systems". Again, this feature is **not** disclosed anywhere in Heil '374, the Intelligent I/O (I_2O) Architecture Specification, or Bonola '279.

Dependent claim 27 further defines that the "Local Transport further has a send handler function and [the] Remote Transport further as a receiver handler function which are respective program interfaces for receiving an inbound message from a remote server on said system area network for direct access to local IOP and for delivering an outbound message to said remote server on said system area network". Again, these feature are **not** disclosed anywhere in Heil '374, the Intelligent I/O (I_2O) Architecture Specification, or Bonola '279.

Lastly dependent claim 28 further defines that the "Remote Transport further builds an IOP connection structure including at least an IOP descriptor pointer which refers to the IOP descriptor structure of the Connection Manager for making a direct call to the Local Transport through the receiver handler function and the send handler function". Again, these feature are **not** disclosed anywhere in Heil '374, the Intelligent I/O (I_2O) Architecture Specification, or Bonola '279.

In view of the foregoing explanations, and in view of the fact that neither Heil '374, the Intelligent I/O (I_2O) Architecture Specification, nor Bonola '279, whether taken in combination or individually, discloses and suggests Appellants' dependent claims 24-28. Therefore Appellants respectfully request that the rejection of dependent claims 24-28 be reversed as well.

IX. Conclusion

In view of the law and the facts presented herein, Appellants respectfully submit that the

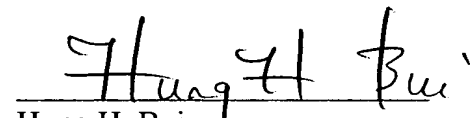
Examiner's proposed combination of Heil '374, the Intelligent I/O (I_2O) Architecture Specification, and Bonola '279, fails to disclose or suggest Appellants' claimed invention. The §103 art rejections are essentially based on the erroneous belief that Heil '374, the Intelligent I/O (I_2O) Architecture Specification, and Bonola '279 discloses all the claimed features. However, the Appellants have established this as incorrect. None of prior art references was correctly relied by the Examiner to support the §103 rejections. As stated in In re Schaefer, 229 F.2d 476, 108 U.S.P.Q. 326 (C.C.P.A 1956):

> "[T]o determine whether the combination of references is improper, the following criterion is often used: namely, whether the prior art suggests doing what applicant has done ... [I]t is not enough for a valid rejection to review the prior art in retrospect once an applicant's disclosure is known. The art applied should be viewed by itself to see if it fairly disclosed doing what an applicant has done." *See also* In re Taborski, 556 F.2d 85, 183 USPQ 350 (CCPA 1973); In re Regal, 526 F.2d 1399, 188 USPQ 136 (CCPA 1976); and In re Imperato, 46 F.2d 585, 179 USPQ 730 (CCPA 1973).

Here, it is abundantly clear that all cited prior art references simply do not disclose or suggest what Appellants have done. For all of the foregoing reasons, Appellants respectfully request the Board reverse the final rejection of claims 1-28.

No fee is incurred by the Appeal Brief since such a fee has already been paid on November 13, 2001. However, please charge any shortage of fees due in connection with the filing of this paper, including extension of time fees, if necessary, to the Deposit Account of Antonelli, Terry, Stout & Kraus, No. 01-2135 (Application No. 219.36435X00), and please credit any excess fees to said deposit account.

Respectfully submitted,
Antonelli, Terry, Stout & Kraus, L.L.P.



Hung H. Bui
Attorney for the Appellants
Registration No.: 40,415

Date: **December 6, 2002**
Antonelli, Terry, Stout & Kraus, LLP
1300 North Seventeenth Street
Suite 1800
Arlington, VA 22209
(703) 312-6600 (phone)
(703) 312-6666 (fax)

Appendix:

Claims 1-21 and 23-28 On Appeal

1. (Twice Amended) An input/output platform (IOP) access module installed in a host system for providing input/output device access between the host system and another system, via a data network, said IOP access module comprising:

a Local Transport arranged to provide an interface to an input/output platform (IOP) supporting an array of input/output devices;

a Remote Transport arranged to provide an interface to said another system, via said data network; and

a Connection Manager arranged to establish connection services and to create a direct call path between the Local Transport and the Remote Transport so as to provide access to input/output devices.

2. The input/output platform (IOP) access module of claim 1, wherein said IOP access module is one of a hardware module, a combined hardware/software module, and a software module provided on a tangible medium.

3. (Amended) The input/output platform (IOP) access module of claim 1, wherein said host system corresponds to a host server, said another system corresponds to any one of remote servers, via said data network.

4. The input/output platform (IOP) access module of claim 2, further comprising said IOP which comprises:

at least one or more input/output processors;

at least one storage device as said input/output devices;

a device driver module arranged to interface with said storage device;

a communication layer which defines a mechanism for communications between the Local Transport and the device driver module.

5. The input/output platform (IOP) access module of claim 4, wherein said communication layer is responsible for managing all service requests and providing a set of Application Programming Interfaces (APIs) for delivering messages, along with a set of support routines that process the messages.

6. The input/output platform (IOP) access module of claim 5, wherein said communication layer comprises a message layer which sets up a communication session, and a transport layer which defines how information will be shared.

7. (Twice Amended) A host system, comprising:

a processor including an operating system (OS);

an array of storage devices;

a driver module for exporting local storage device access onto a computer network, said driver module comprising:

a device driver module arranged to provide an interface to said array of local storage devices;

a host driver module arranged to provide an interface to the operating system (OS), said host driver module comprising a Local Transport which communicates with the device driver module, a Remote Transport which provides an interface to said computer network, and a Connection Manager which establishes connection services with remote systems on said computer network and coordinates functions responsible for creating a direct call path between the Local Transport and the Remote Transport to provide access to said storage devices; and

a communication layer which supports communications between the host driver module and the device driver module.

8. The host system of claim 7, wherein said host system corresponds to a host server, said remote systems corresponds to remote servers arranged in a cluster, and said computer network corresponds to a system area network for communications between said host system and said remote systems within said cluster.

9. The host system of claim 7, wherein said communication layer is responsible for managing all service requests and providing a set of Application Programming Interfaces (APIs) for delivering messages, along with a set of support routines that process the messages.

10. The host system of claim 9, wherein said communication layer comprises a message layer which sets up a communication session, and a transport layer which defines how information will be shared.

11. (Amended) The host system of claim 9, wherein said host driver module and said device driver module constitute a single device that is portable across a plurality of operating

systems and host network platforms, and works interoperably with a plurality of storage devices and operating systems.

12. (Twice Amended) The host system of claim 9, wherein said host driver module and said device driver module operate in accordance with an Intelligent Input/Output (I_2O) specification for allowing storage devices to operate independently from the operating system (OS).

13. The host system of claim 7, wherein said driver module is one of a hardware module, a combined hardware/software module, and a software module provided on a tangible medium.

14. A driver configuration of a host server for exporting local storage device access onto a computer network, comprising:

an input/output platform (IOP) arranged to control an array of local storage devices; and

a system driver module comprising:

a Local Transport arranged to provide an interface to said input/output platform (IOP);

a Remote Transport arranged to provide an interface to said computer

network; and

a Connection Manager arranged to establish connection services with remote servers on said computer network and coordinate functions responsible for creating a direct call path between the Local Transport and the Remote Transport to provide access to the local storage devices.

15. The driver configuration of claim 14, wherein said input/output platform (IOP) supports at least one or more input/output processors, and comprises:

a device driver module which interfaces the local storage devices for controlling said array of local storage devices; and

a communication layer which defines a mechanism for communications between the system driver module and the device driver module.

16. The driver configuration of claim 15, wherein said communication layer is responsible for managing and dispatching all service requests and providing a set of Application Programming Interfaces (APIs) for delivering messages, along with a set of support routines that process the messages, and is comprised of a message layer which sets up a communication session, and a transport layer which defines how information will be shared.

17. The driver configuration of claim 15, wherein said system driver module and said device driver module constitute a single device that is portable across a plurality of host operating systems and host network platforms, and works interoperably with a plurality of storage devices and host operating systems.

18. The driver configuration of claim 15, wherein said system driver module and said device driver module operate in accordance with an Intelligent Input/Output (I_2O) specification for allowing storage devices to operate independently from the operating system of the host server.

19. The driver configuration of claim 15, wherein, upon initialization, said Local Transport scans the local bus so as to locate and initialize all local input/output platforms (IOPs) and builds an opaque "context" structure for each input/output platform (IOP), wherein said Remote Transport prepares to accept requests from a remote server through said computer network, and wherein said Connection Manager queries said Local Transport so as to determine the number of input/output platforms (IOPs), builds an IOP descriptor structure for each input/output platform (IOP) which includes an exported table of function call pointers and the context required by the Local Transport to communicate with the input/output platform (IOP), and finally establishes a

network management communication channel through the Remote Transport, which waits for an external connection from said remote server on said computer network for exporting local device access onto said computer network using said direct call path between the Local Transport and the Remote Transport.

20. The driver configuration of claim 19, wherein said Local Transport further has a send handler function and said Remote Transport further has a receive handler function which are respective program interfaces for receiving an inbound message from a remote server on said computer network for direct access to local input/output platform and for delivering an outbound message to said remote server on said computer network.

21. The driver configuration of claim 19, wherein said Remote Transport further builds an IOP connection structure including at least an IOP descriptor pointer which refers to the IOP descriptor structure of the Connection Manager for making a direct call to the Local Transport through the receive handler function and the send handler function.

23. (Amended) A process of establishing a service connection to a local input/output platform (IOP) connected to a local bus using a system driver module in response to a

request from a remote server on a system area network, comprising the steps of:

beginning initialization of said driver module which provides access to a local storage system while bypassing protocol stacks of a host operating system, said system driver module comprising a Local Transport which provides direct access to the local storage device system, a Remote Transport which interfaces to other nodes of said system area network, and a Connection Manager which provides connection services and coordinates functions responsible for creating a direct call path between the Local Transport and the Remote Transport;

scanning, at said Local Transport, the local bus to locate and initialize all local input/output platforms (IOPs), and building an IOP context structure for each input/output platform (IOP) found;

preparing, at said Remote Transport, to accept a request for a service connection from said remote server on said system area network;

asking, at said Connection Manager, whether said Local Transport determines the number of input/output platforms (IOPs), and building a descriptor structure for each input/output platform (IOP) which includes an exported table of function call pointers and the context required by the Local Transport to communicate with the input/output platform (IOP); and

establishing a system area network management communication channel through the Remote Transport, which waits for an external connection from said remote server on said system area network for exporting local device access onto said system area network using said direct call

path between the Local Transport and the Remote Transport.

24. The process of claim 23, wherein said input/output platform (IOP) comprises:

a device driver module which interfaces the local storage devices, and which controls an array of local storage devices; and

a communication layer which defines a mechanism for communication between the system driver module and the device driver module.

25. The process of claim 24, wherein said communication layer is responsible for managing and dispatching all service requests and providing a set of Application Programming Interfaces (APIs) for delivering messages, along with a set of support routines that process the messages, and is comprised of a message layer which sets up a communication session, and a transport layer which defines how information will be shared.

26. The process of claim 23, wherein said system driver module and said device driver module constitute a single device that is portable across a plurality of host operating systems and host network platforms, and operate in accordance with an Intelligent Input/Output (I_2O) specification for allowing storage devices to operate independently from the host operating system.

27. The process of claim 23, wherein said Local Transport further has a send handler function and said Remote Transport further has a receive handler function which are respective program interfaces for receiving an inbound message from a remote server on said system area network for direct access to local input/output platform (IOP) and for delivering an outbound message to said remote server on said system area network.

28. The process of claim 27, wherein said Remote Transport further builds an IOP connection structure including at least an IOP descriptor pointer which refers to the IOP descriptor structure of the Connection Manager for making a direct call to the Local Transport through the receive handler function and the send handler function.